2-28-00 A



*IBM Docket No.* RAL9-99-0073



# In the United States Patent and Trademark Office
## Patent Application Transmittal

***Transmitted herewith for filing is the Patent Application of:***

Inventors(s): Cedell Adam Alexander, Jr., Loren Douglas Larsen

For: Portable Networking Interface Method and Apparatus for Distributed Switching System

***Enclosed are***

*28* pages of specification, including *34* claims, plus *7* sheets of drawings.

X An assignment of the invention to International Business Machines Corporation, Armonk, New York 10504.

A certified copy of a/an application.

X Declaration and Power of Attorney.

x PTO-1449 & references

x A return post card

Other:

***Filing Fee Calculation (For Other Than Small Entity)***

| | | | | | | |
|---|---|---|---|---|---|---|
| Basic Fee: | | | | | | $690.00 |
| Claims Fees: | Filed | Limit | Extra | | Rate per Extra | |
| Total claims: | 34 | 20 | 14 | | $18.00 | $252.00 |
| Independent claims: | 5 | 3 | 2 | | $78.00 | $156.00 |
| | Multiple Dependent Claim Presented | | | | $260.00 | $0.00 |
| | | | | | **Total** | **$1,098.00** |

Please charge Deposit Account **09-0464** for the **Total** set forth above. The Commissioner is authorized to charge payment of any additional filing fees required under 37 CFR §1.16 and any patent application processing fees under 37 CFR §1.17 or to credit any overpayment to the identified account. A duplicate copy of this sheet is enclosed.

*Express Mail Certificate*

Express Mail Label No: EJ113457344US

Date: Feb. 25, 2000

I hereby certify that I am depositing the papers identified above with the U.S. Postal Service "Express Mail Post Office to Address" service on the above date, addressed to the Commissioner of Patents and Trademarks, Washington, DC 20231

Linda Dupont

Linda Dupont

BY: Joscelyn G. Cockburn

Joscelyn G. Cockburn

Attorney of Record Reg. No. 27,069

Date: Feb. 25, 2000

IBM Corporation 972/B656
Intellectual Property Law
PO Box 12195
Res. Tri. Park, NC 27709

Telephone: 919-543- 9036 FAX 919-543-3634

# PORTABLE NETWORKING INTERFACE METHOD AND APPARATUS FOR DISTRIBUTED SWITCHING SYSTEM

## TECHNICAL FIELD

The present invention relates in general to a switching system for use in a network. More particularly, the invention relates to a portable interface method and system for accessing a switch device driver from the various network services applications supported by a switch.

## BACKGROUND INFORMATION

The proliferation of personal computers, digital telephones, telephony and telecommunications technology has resulted in the development of complex switches in order to efficiently communicate digital data between a number of different devices. These communication systems are generally referred to as networks. Each network operates on the basis of one or more switches which route digital data from an originating device to a destination device. To this end, communication protocols have been developed in order to standardize and streamline communications between devices and promote connectivity.

As advances are made in telecommunications and connectivity technology, additional protocols are rapidly being developed in order to improve the efficiency and interconnectivity of networking systems. As these advances occur, modifications are required to the switches in order to allow the switches to appropriately deal with the new protocols and take advantage of the new efficiencies that they offer.

Unfortunately, a switch can represent a large capital investment in a network system. The frequency in which new protocols are developed makes it impractical to upgrade switches with every protocol introduced to the market. Accordingly, what is needed is a system and device for improving interface portability within the switch so that switches can be quickly and easily upgraded and new network interface protocols can be written and supported on multiple switch fabrics.

## SUMMARY OF THE INVENTION

The invention solves the problem of portability by defining two primary interfaces within the switch. The first interface is called the Forwarding Database Distribution Library (FDDL) Application Program Interface (API). The primary purpose of this interface is to allow each protocol application to distribute its database and functionality to intelligent port controllers within the switch. Such distribution facilitates hardware forwarding at the controller. Each protocol application may define a specific set of FDDL messages that are exchanged between the protocol application and the switch fabric, which passes the messages to software running at each port controller.

The second interface defined by the invention is called the Switch Services API. This interface is primarily a generic way for controlling data message flow between the ports interfaces and the switch device driver. A set of specific messages is defined to allow uniform exchange of information about the hardware status of the port as well as an interface for sending and receiving data frames.

The forgoing broadly outlines the features and technical advantages of the present invention in order that the detailed description of the invention that follows may be better understood. Additional features and advantages of the invention will be described hereafter, which form the basis of the claims of the invention.

## BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanied drawings, in which:

**Fig. 1** is a system block diagram of a network switch, including workstations connected to the network switch;

**Fig. 2** is a system block diagram of a data processing system which may be used as a workstation within the present invention;

**Fig. 3** is a block diagram describing the FDDL defined by the present invention and its relationship with the switch device driver and protocol drivers;

**Fig. 4** is a software system block diagram of a portion of a network switch embodying the present invention which describes the relationship between the FDDL, the other services provided by the switch, and the in relation to the switch device driver;

**Fig. 5** is a system block diagram of the software architecture within a network switch embodying the present invention;

**Fig. 6** is a flow chart according to ANSI/ISO Standard 5807-1985 depicting the operation of the basic primitives defined by the Switch Services API of the instant invention; and

**Fig. 7** is a flow chart according to ANSI/ISO Standard 5807-1985 demonstrating the operation of the FDDL API as defined by the instant invention.

## DETAILED DESCRIPTION OF THE INVENTION

In the following description, numerous specific details are set forth such as languages, operating systems, microprocessors, workstations, bus systems, networking systems, input/output (I/O) systems, etc., to provide a thorough understanding of the invention. However, it will be obvious to those skilled in the art that the present invention may be practiced without such specific details. In other instances, well-known circuits, computer equipment, network protocols, programming configurations, or wiring systems have been shown in blocked diagram form in order to not obscure the present invention in unnecessary detail. For the most part, details concerning timing considerations, specific equipment used, specific programming languages and protocols used, specific networking systems used, and the like have been omitted in as much as these details are not necessary to obtain a complete understanding of the present invention and are well within the skills of persons of ordinary skill in the art.

The switch to which the present invention relates is shown with reference to **Fig. 1**. A network switch **100** is comprised of one or more intelligent port controllers **110**, a switch fabric **112**, and a central processing unit (CPU) **114**. The switch **100** is connected to one or more backbones **104**, which in turn are connected to one or more workstations **102**. Each intelligent port controller **110** may be connected to one or more backbones **104** comprising a local area network (LAN) **106**. The entire system may be referred to as a network **108**.

The switch fabric **112** is comprised of one or more processors that manage a shared pool of packet/cell memory. The switch fabric **112** controls the sophisticated queuing and scheduling functions of the switch **100**.

The intelligent port controller **110** provides connectivity between the switch fabric **112** and the physical layer devices, such as the backbones **104**. The intelligent port controller **110** may be implemented with one or more bitstream processors.

A typical workstation **102** is depicted with reference to **Fig. 2**, which illustrates the typical hardware configuration of workstation **213** in accordance with the subject invention. The workstation **213** includes a central processing unit (CPU) **210**, such as a conventional microprocessor and a number of other units interconnected via a system bus **212**. The workstation **213** may include a random access memory (RAM) **214**, a read-only memory (ROM) **216**, and an I/O adapter **218** for connecting peripheral devices, such as disk units **220** and tape drives **240** to the bus **212**. The workstation **213** also include a user interface adapter **222** for connecting a keyboard **224**, a mouse **226** and/or other user interface devices, such as a touch screen device (not shown) to the bus **212**, a communication adapter **234** for connecting the workstation **213** to a network **242** (such as the one depicted on **Fig. 1** at **108**), and a display adapter **236** for connecting the bus **212** to a display device **238**. The CPU **210** may include other circuitry not shown, which may include circuitry found within a microprocessor, *e.g.,* execution unit, bus interface unit, arithmetic logic unit (ALU), etc. The CPU **210** may also reside on one integrated circuit (IC).

The FDDL is defined with reference to **Fig. 3**. The FDDL is a library which defines a set of API's designed to enable protocol forwarding functions to be distributed in a manner that is simple, efficient, and deportable. The FDDL **310** is comprised of one or more towers **322**, **324, 326, 328**. As depicted, a tower may be provided for remote monitoring (RMON) in an RMON FDDL tower **322**. Multi Protocol Over ATM (MPOA) services may be provided through an MPOA client FDDL tower **324**. Bridging services may connect through a Bridge FDDL tower **326**. Internet Protocol (IP) Autolearn connectivity may be provided through an Autolearn FDDL tower **328**.

Each of the FDDL towers **322, 324, 326, 328** is connected through the FDDL API **332** to its respective protocol services of the RMON application **314**, the MPOA application **316**, the Bridge **318**, and the IP Autolearn application **320**, as provided within the switch.

The FDDL **310** functions to receive commands from the various protocol components **314, 316, 318, 320** into the corresponding FDDL towers **322, 324, 326, 328**. When a command is received into a tower **322, 324, 326, 328**, it is passed to the base FDDL subsystem **330** for translation and passage directly to the switch device driver **312** through the Switch Services API **334**.

The operation of the Switch Services API is demonstrated with reference to **Fig. 4.** The switch device driver **420** resides immediately below FDDL in the CPU protocol stack. As shown, there may be several users of the Switch Service API **410** which communicate with the switch device driver **420**. In addition to the FDDL towers **418**, other users may include an Ethernet Device Driver Shim **416**

and an Asynchronous Transfer Mode (ATM) Device Driver Shim **414**. The Device Driver Shims **414, 416** are interface translation agents which complete the high-level of architecture of the switch. The shims translate between the existing device driver interfaces and the Switch Services API **410** of the instant invention. In this way, translation through the shims **414, 416** allows preservation of the existing device driver interfaces from the ATM and Ethernet protocols and avoids modification of those handlers for use with the switch services API **410**.

The bridging protocol application **412** may also communicate directly with the switch device driver **420** through the Switch Services API **410**.

The architecture into which the FDDL and APIs of the instant invention fit is demonstrated with reference to **Fig. 5**, which is a block diagram depicting the basic software architecture of a network switch embodying the instant invention. While the software depicted is depicted as running on a Power PC processor **510** and on the OS Open real time operating system **512**, those skilled in the art will appreciate that the instant invention can be practiced with a number of processors running a number of different operating systems. However, since the Power PC platform is the preferred technology for products employing many of the networking technology described, it present many advantages with regard to the architectural goals of the instant invention.

The Power PC box **518** is connected to the switch fabric **514** through the Switch Device Driver **516**. In turn, the switch fabric **514** is connected to one or more port controllers **520**. The Switch Device Driver **516** supports a Switch Services API **522** through which it can send and receive messages to the FDDL

**524**, as well as the ATM Device Driver Shim **526** and the Ethernet Device Driver Shim **528**. The ATM Device Driver Shim **526** and the Ethernet Device Driver Shim **528** connect to their respective net handlers **530, 532** through device driver interfaces **534, 536**.

The MPOA client **538** may communicate to the switch device driver either through the ATM API **540** or through the FDDL API **542** as defined by the FDDL **524**. The bridge services **544**, including the Virtual LAN (VLAN) and IP Autolearn services may be provided through the Ethernet Net Handler **532**, through the FDDL API **542** to the FDDL **524**, or LAN Emulation Client (LEC) **546** may be provided to communicate through the ATM API **540** to the ATM Net Handler **530**.

Through the structure defined, the operating system **512** features such as Simple Network Management Protocol (SNMP) and RMON **548**, other box services **550**, and IP hosting services **552**, such as Telnet, Ping, and other may be provided.

The operation of the Switch Services API **522** as provided by the switch device driver **516** is shown with reference to **Fig. 6**. Execution begins **610** without precondition. The API is initiated with a switch _registration() call **612** to register an interface user of the Switch Services API. The registration call includes parameters of a code point identifying the interface application that is registering with the API and pointers to up-call functions which may be called when messages or data frames associated with the application are received by the switch device driver to be passed through the API.

Once the switch _registration() called **612** is made, the API is active **614**. While the API is active, calls may be made to at least any one of four primitives, including switch _send _MSG() **616**, switch _send _data () **618**, switch _get _buffer () **620,** and switch _free _buffer () **622**.

The switch _send _MSG() primitive **616** is called to transmit a message to one or more registered interfaces. Messages may be sent to one interface, a group of interfaces, or broadcast to all interfaces. A message may be generally formatted using the Type-Length-Value (TLV) convention.

The switch _ send_ data() primitive **618** is called to transmit a data frame out of one or more interfaces. When a frame is to be transmitted to more than one interface, the set of destination interfaces may be specified with a bit mask or by other means well-appreciated within the art.

The switch_get_buffer() primitive **620** is called to allocate frame buffers. Conversely, the switch _free_buffer() primitive **622** is called to deallocate frame buffers.

Calls to the primitives may continue as long as the API is active **624**. When an interface application wishes to disable the API, it does so by calling switch_deregistration() **626**, which deregisters the application as a user of the switch services API. Execution of the Switch Services API then ceases **628**.

The operation of the base FDDL subsystem is demonstrated with reference to **Fig.** 7. Execution begins **710** without pre-condition. The FDDL _registration() primitive **712** is called to register a client application as a user of the FDDL API. A call to the FDDL_registration() primitive **712** specifies a code point identifying

the data base of the calling application (*e.g.* bridging, MPOA, etc.) and provides a pointer to a message-reception call-back function that can be invoked when messages related to the specified client are received by the API.

After the primitive FDDL_registration() **712** is called, the FDDL is active **714**, beginning a looping process of calls.

Within the loop, the FDDL_send() primitive **716** may be called to initiate transmission of a message from the CPU to one or more adapters. The message may be transmitted to a single adapter or broadcast to all adapters. The FDDL_registration_status()primitive **718** may be called query whether a particular database is currently registered with the FDDL API.

When it is no longer desired for the FDDL to be active **720**, the primitive_deregistration() **722** may be called to deregister a client application as a user of the FDDL API. Following the call to the FDDL_deregistration() **722**, execution of the FDDL subsystem ceases **724**.

It will be well appreciated by those skilled in the art that each of the FDDL towers as shown on **Fig. 3**, including the RMON tower **322**, the MPOA tower **324**, the Bridge tower **326**, and the IP Autolearn tower **328** may each be optimized with primitives adapted to their respective applications **314**, **316**, **318**, **320**. Those skilled in the art will also appreciate that primitives need not be written for each tower and that additional towers may be added for client applications to be added in the future. However, the base FDDL subsystem **330** and its primitives may remain unchanged in order to provide a universal interface to the switch device driver **312**.

The FDDL towers **322**, **324**, **326**, **328** may each have its own registration processes that allow instances of its specific protocol client applications to register. Additionally, those skilled in the art will appreciate that the FDDL tower calls may be providing for other networking features well-known in the art, such as providing reliable delivery of messages, acknowledgment and non-acknowledgment schemes, Cyclic Redundancy Code (CRC) code checking, and the like.

Those skilled in the art will also appreciate that the Switch Services API need not provide for such flexibility. The Switch Device Drivers **312** are hardware dependent relying on the switch fabric (**Fig. 5, 514**) for their definition. As hardware will not be replaced or upgraded as easily or frequently as the client applications, the Switch Services API need not provide a towering structure.

As to the manner of operation and use of the instant invention, the same is made apparent from the foregoing discussion. With respect to the above description, it is to be realized that although embodiments of specific material, representations, primitives, languages, and network configurations are disclosed, those enabling embodiments are illustrative and the optimum relationship for the parts of the invention is to include variations in composition, form, function, and manner of operation, which are deemed readily apparent to one skilled in the art in view of this disclosure. All relevant relationships to those illustrated in the drawings in this specification are intended to be encompassed by the present invention.

Therefore, the foregoing is considered as illustrative of the principles of the invention, and since numerous modifications will occur to those skilled to those in

the art, it is not desired to limit the invention to exact construction and operation shown or described, and a user my resort to all suitable modifications and equivalence, falling within the scope of the invention.

WHAT IS CLAIMED IS:

1. A network switch comprising

a CPU;

a memory system having circuitry operable to attach to the CPU;

a switch fabric system having circuitry operable to attach to the CPU;

a port controller having circuitry operable to attach to the switch fabric system;

a software application operable to execute on the CPU;

a Forwarding Database Distribution Library (FDDL) system operable to execute on the CPU; and

a switch device driver operable to execute on the CPU,

wherein the software application is operable to communicate with the FDDL system, the FDDL system is operable to communicate with the switch device driver, and the switch device driver is operable to communicate with the switch fabric.

2. The network switch of claim 1 further comprising a second software application operable to execute on the CPU,

wherein the second software application communicates with the FDDL system.

3. The network switch of claim 1 wherein the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

4. The network switch of claim 2 wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

5. The network switch of claim 2 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system

wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

6. The network switch of claim 1 further comprising:

an independent software application operable to execute on the CPU; and

an independent software application shim operable to execute on the CPU,

wherein the independent software application communicates with the independent software application shim and the independent software application shim communicates with the switch device driver.

7. The network switch of claim 6 further comprising a second software application operable to execute on the CPU,

wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

8. The network switch of claim 6 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system

wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software

application tower FDDL system and the second software application tower FDDL system.

9. A network switch comprising:

a CPU;

a memory system having circuitry operable to attach to the CPU;

a switch fabric system having circuitry operable to attach to the CPU;

a port controller having circuitry operable to attach to the switch fabric system;

a protocol means for providing a service to a network system;

a Forwarding Database Distribution Library (FDDL) means for communicating with the protocol means; and

a switch device driver means for communicating with the FDDL means and the port controller.

10. The network switch of claim 9 further comprising a second protocol means for providing a second service to the network system,

wherein the FDDL means communicates with the second protocol means.

11. The network switch of claim 9 wherein the FDDL means defines an FDDL API for communication with the software application, and the FDDL means defines a Switch Services API for communication with the switch device driver.

12. The network switch of claim 10 wherein the FDDL means defines an FDDL API for communication with the protocol means and the second protocol means, and the FDDL system defines a Switch Services API for communication with the switch device driver means.

13. The network switch of claim 10 wherein the FDDL means comprises:

a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with the second protocol means and the base FDDL means.

14. The network switch of claim 9 further comprising:

an independent protocol means for providing an independent service to the network system; and

an independent protocol shim for communicating with the independent protocol means and the switch device driver means.

15. The network switch of claim 14 further comprising a second protocol means for providing a second service to the network system,

wherein the FDDL means communicates with the second protocol means.

16. The network switch of claim 14 wherein the FDDL means comprises:

a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with the second protocol means and the base FDDL means.

17. A method of providing communications over a network system utilizing a first protocol and a second protocol, the method comprising the steps of:

receiving information at a port controller in a first protocol from a first node machine;

communicating the information from the port controller to a switch fabric;

communicating the information from the switch fabric to a switch device driver within an operating system;

communicating the information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the information from the FDDL to a first protocol client.

18. The method of claim 17 further comprising the steps of:

receiving additional information at a port controller in a second protocol from a first node machine;

communicating the additional information from the port controller to a switch fabric;

communicating the additional information from the switch fabric to a switch device driver within an operating system;

communicating the additional information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the additional information from the FDDL to a second protocol client.

19. The method of claim 18 wherein

all communicating between the switch device driver to the FDDL is done through a switch services API; and

all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.

20. The method of claim 18 further comprising the steps of:

defining a switch services API for communication between the switch device driver; and

defining an FDDL API for communication between the first protocol client and the FDDL.

21. The method of claim 18 further comprising the steps:

receiving the information from the switch device driver at an FDDL base within the FDDL;

passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and

sending the information from the first protocol FDDL tower to the first protocol client.

22. A computer-readable medium having stored thereon computer-executable instructions for performing the steps comprising:

receiving information at a port controller in a first protocol from a first node machine;

communicating the information from the port controller to a switch fabric;

communicating the information from the switch fabric to a switch device driver within an operating system;

communicating the information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the information from the FDDL to a first protocol client.

23. The computer-readable medium of claim 22 having further stored thereon computer-executable instructions for performing the steps comprising:

receiving additional information at a port controller in a second protocol from a first node machine;

communicating the additional information from the port controller to a switch fabric;

communicating the additional information from the switch fabric to a switch device driver within an operating system;

communicating the additional information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the additional information from the FDDL to a second protocol client.

24. The computer-readable medium of claim 23 wherein

all communicating between the switch device driver to the FDDL is done through a switch services API; and

all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.

25. The computer-readable medium of claim 23 having further stored thereon computer-executable instructions for performing the steps comprising:

defining a switch services API for communication between the switch device driver; and

defining an FDDL API for communication between the first protocol client and the FDDL.

26. The computer-readable medium of claim 23 having further stored thereon computer-executable instructions for performing the steps comprising:

receiving the information from the switch device driver at an FDDL base within the FDDL;

passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and

sending the information from the first protocol FDDL tower to the first protocol client.

27. A network system comprising:

a network switch comprising a CPU, a memory system having circuitry operable to attach to the CPU, a switch fabric system having circuitry operable to attach to the CPU a port controller having circuitry operable to attach to the switch fabric system, a software application operable to execute on the CPU, a Forwarding Database Distribution Library (FDDL) system operable to execute on the CPU, and a switch device driver operable to execute on the CPU,

wherein the software application is operable to communicate with the FDDL system, the FDDL system is operable to communicate with the switch device driver, and the switch device driver is operable to communicate with the switch fabric;

a backbone; and

a workstation,

wherein the workstation is logically connected to the backbone, and

wherein the backbone is logically connected to the port controller of the network switch.

28. The network system of claim 27 further comprising a second software application operable to execute on the CPU,

wherein the second software application communicates with the FDDL system.

29. The network system of claim 27 wherein the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

30. The network system of claim 28 wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

31. The network system of claim 28 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system

wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

32. The network system of claim 27 further comprising:

an independent software application operable to execute on the CPU; and

an independent software application shim operable to execute on the CPU,

wherein the independent software application communicates with the independent software application shim and the independent software application shim communicates with the switch device driver.

33. The network system of claim 32 further comprising a second software application operable to execute on the CPU,

wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

34. The network system of claim 32 wherein the FDDL system comprises:

a base FDDL system;

a software application tower FDDL system; and

a second software application tower FDDL system

wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL

system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

# PORTABLE NETWORKING INTERFACE METHOD AND APPARATUS FOR DISTRIBUTED SWITCHING SYSTEM

## ABSTRACT OF THE DISCLOSURE

An apparatus and method to provide a portable networking interface for distributed switching systems. Two Application Program Interfaces (APIs) are defined for communication to a Forwarding Database Distribution Library (FDDL). The FDDL sits between network client applications and the switch device driver in order to provide a uniform interface to the switch device driver. Towers may be added to the FDDL to provide additional functionality specific to certain client applications.

::ODMA\PCDOCS\HOUSTON_1\410639\2
1094:7036-P095US

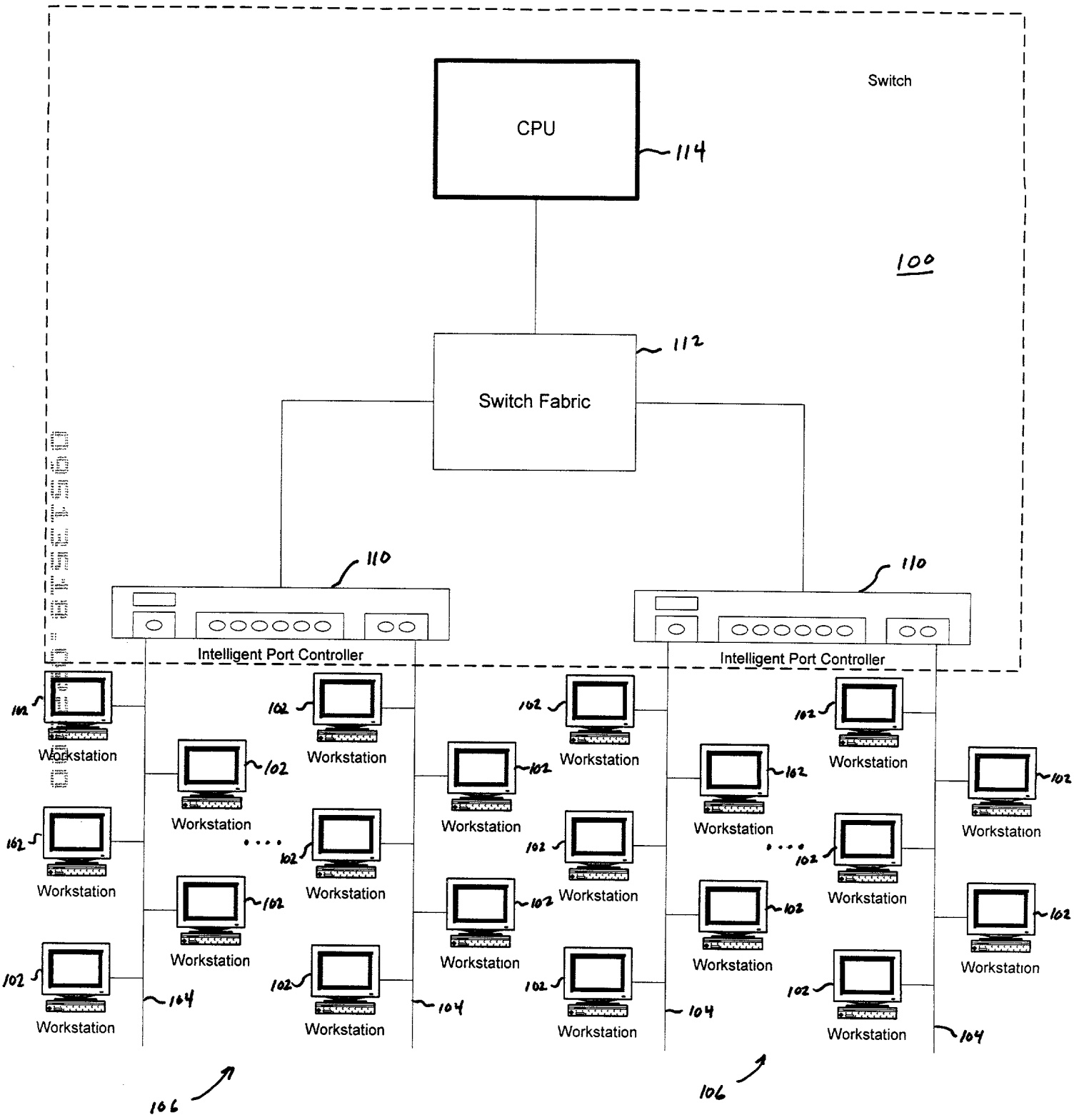
Switch
CPU
114
100
112
Switch Fabric
110
110
Intelligent Port Controller
Intelligent Port Controller
102
Workstation
104
106
106
108

Fig. 1

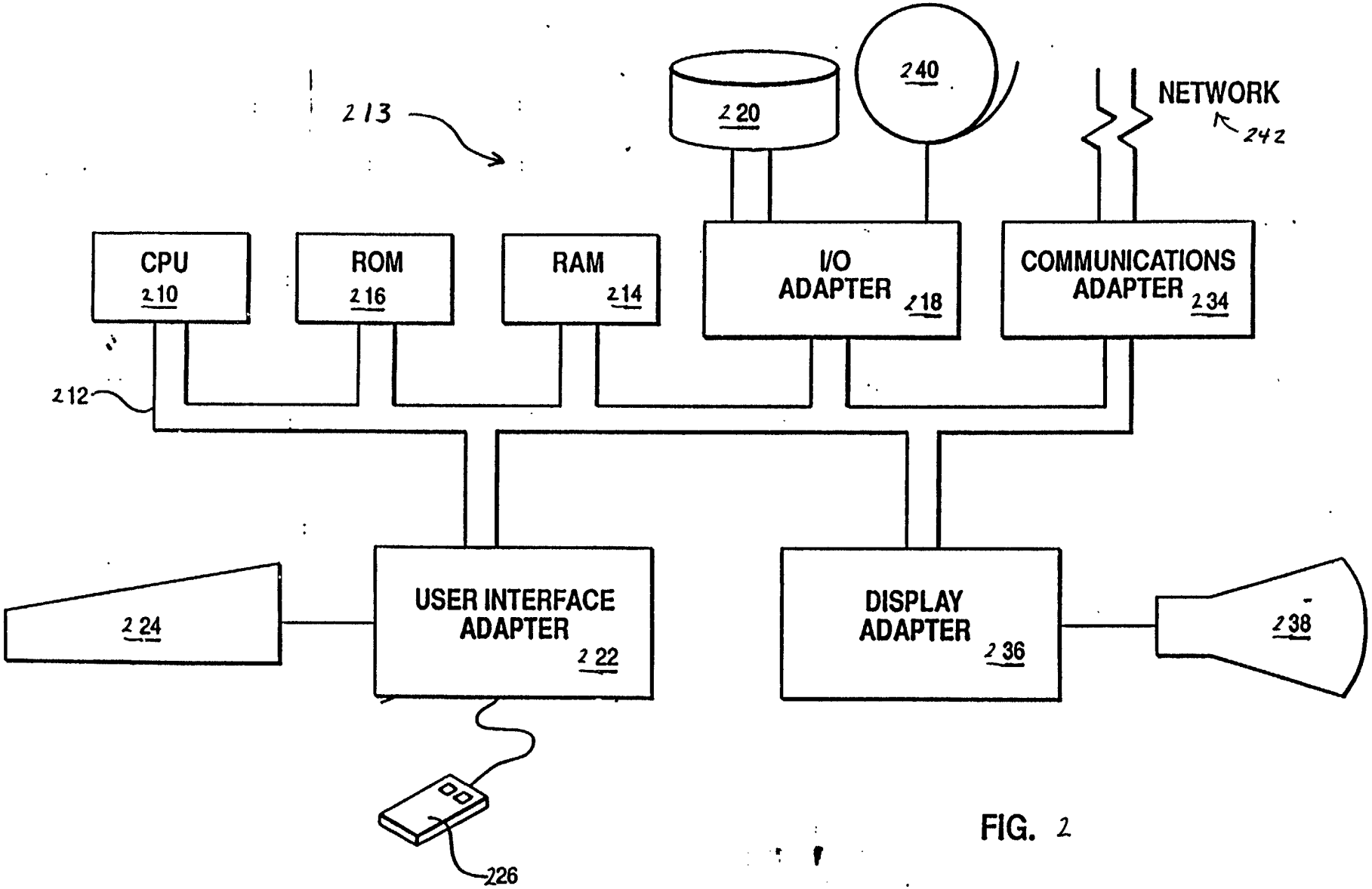


FIG. 2

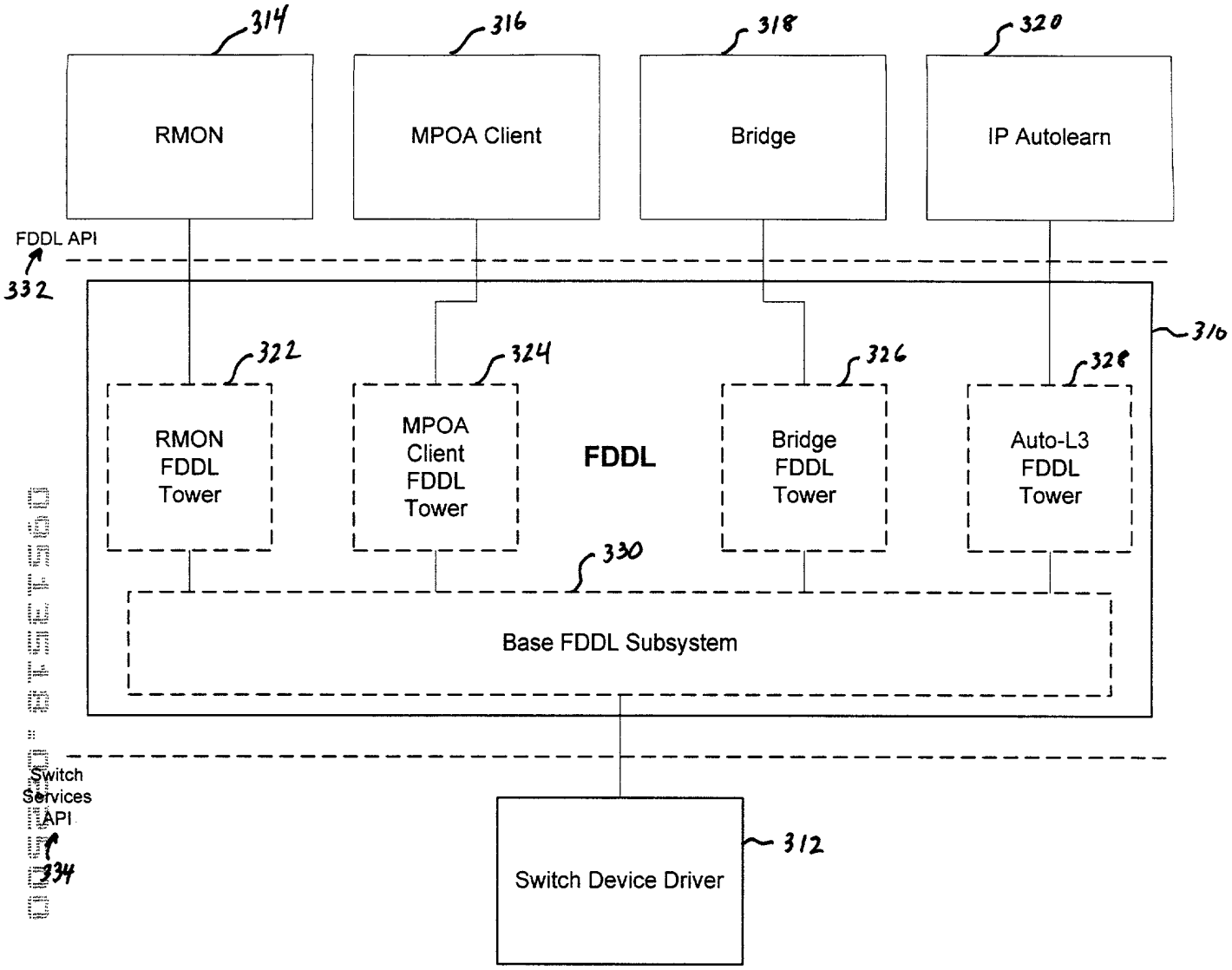
314
316
318
320
RMON
MPOA Client
Bridge
IP Autolearn
FDDL API
332
310
322
324
326
328
RMON FDDL Tower
MPOA Client FDDL Tower
FDDL
Bridge FDDL Tower
Auto-L3 FDDL Tower
330
Base FDDL Subsystem
Switch Services API
334
Switch Device Driver
312

Fig. 3

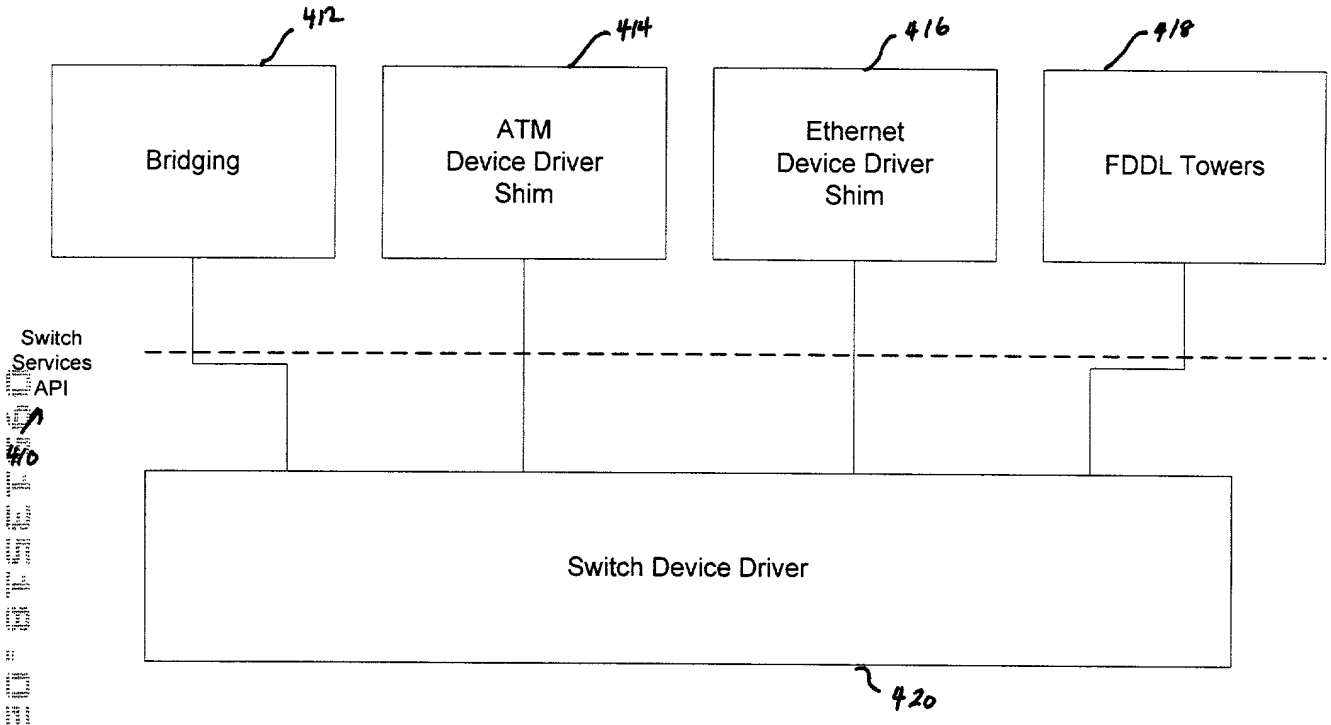
412
414
416
418
Bridging
ATM
Device Driver
Shim
Ethernet
Device Driver
Shim
FDDL Towers
Switch
Services
API
410
Switch Device Driver
420

Fig. 4

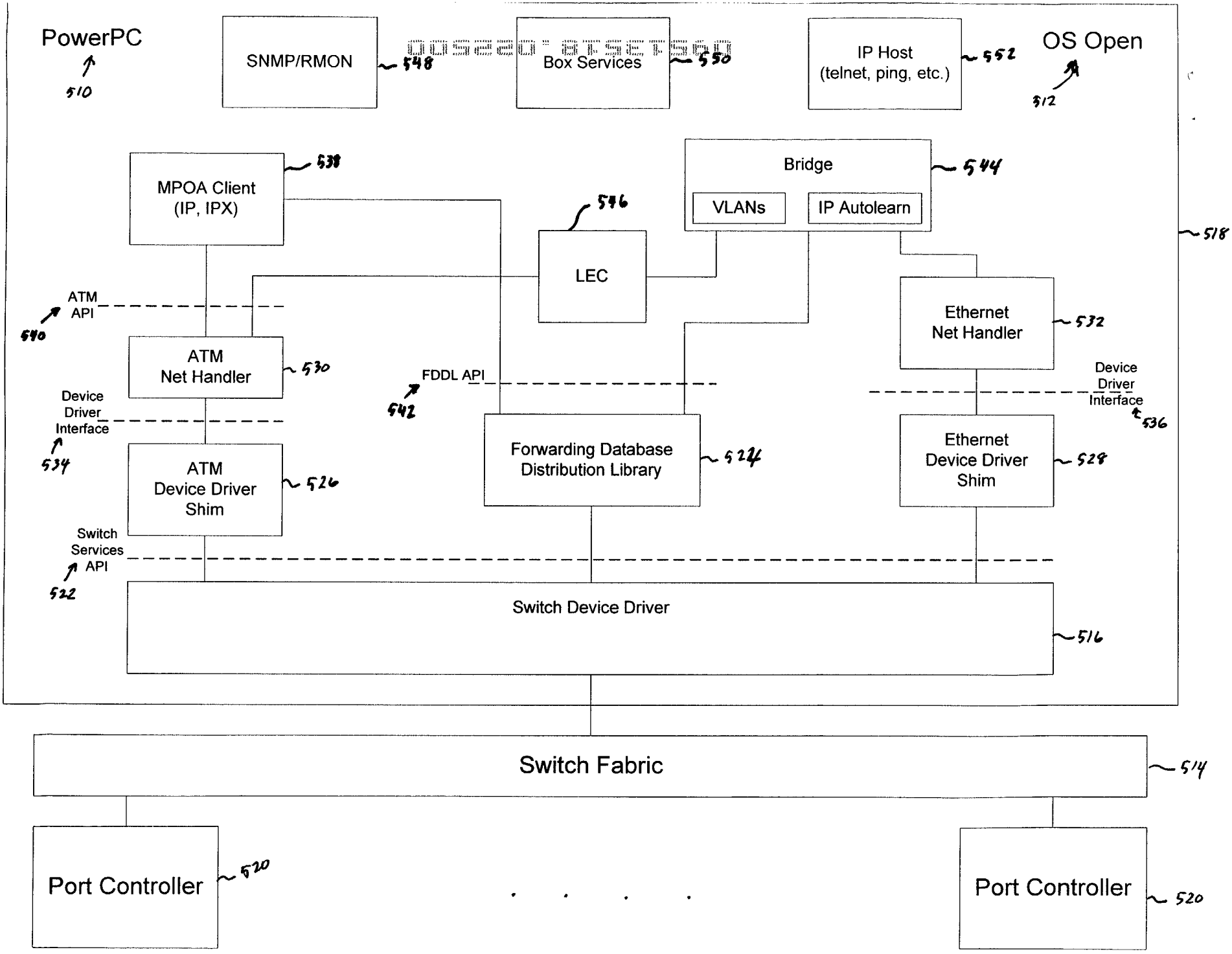
PowerPC
510
SNMP/RMON
548
Box Services
550
IP Host
(telnet, ping, etc.)
552
OS Open
512
MPOA Client
(IP, IPX)
538
Bridge
544
VLANs
IP Autolearn
546
LEC
518
ATM
API
540
ATM
Net Handler
530
Ethernet
Net Handler
532
FDDL API
542
Device
Driver
Interface
534
Device
Driver
Interface
536
ATM
Device Driver
Shim
526
Forwarding Database
Distribution Library
524
Ethernet
Device Driver
Shim
528
Switch
Services
API
522
Switch Device Driver
516
Switch Fabric
514
Port Controller
520
Port Controller
520

Fig. 5

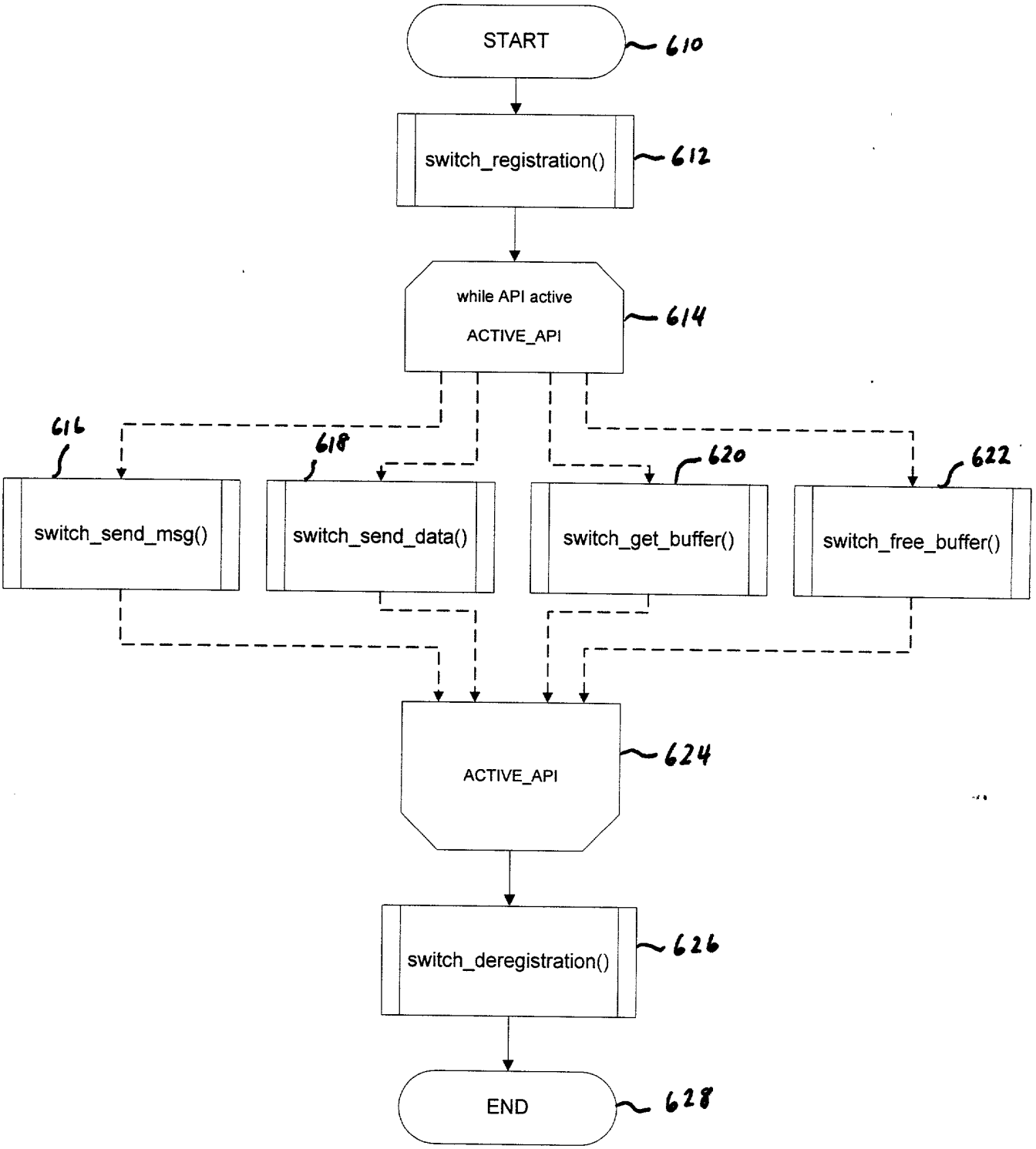
START
610
switch_registration()
612
while API active
ACTIVE_API
614
616
618
620
622
switch_send_msg()
switch_send_data()
switch_get_buffer()
switch_free_buffer()
ACTIVE_API
624
switch_deregistration()
626
END
628

Fig. 6

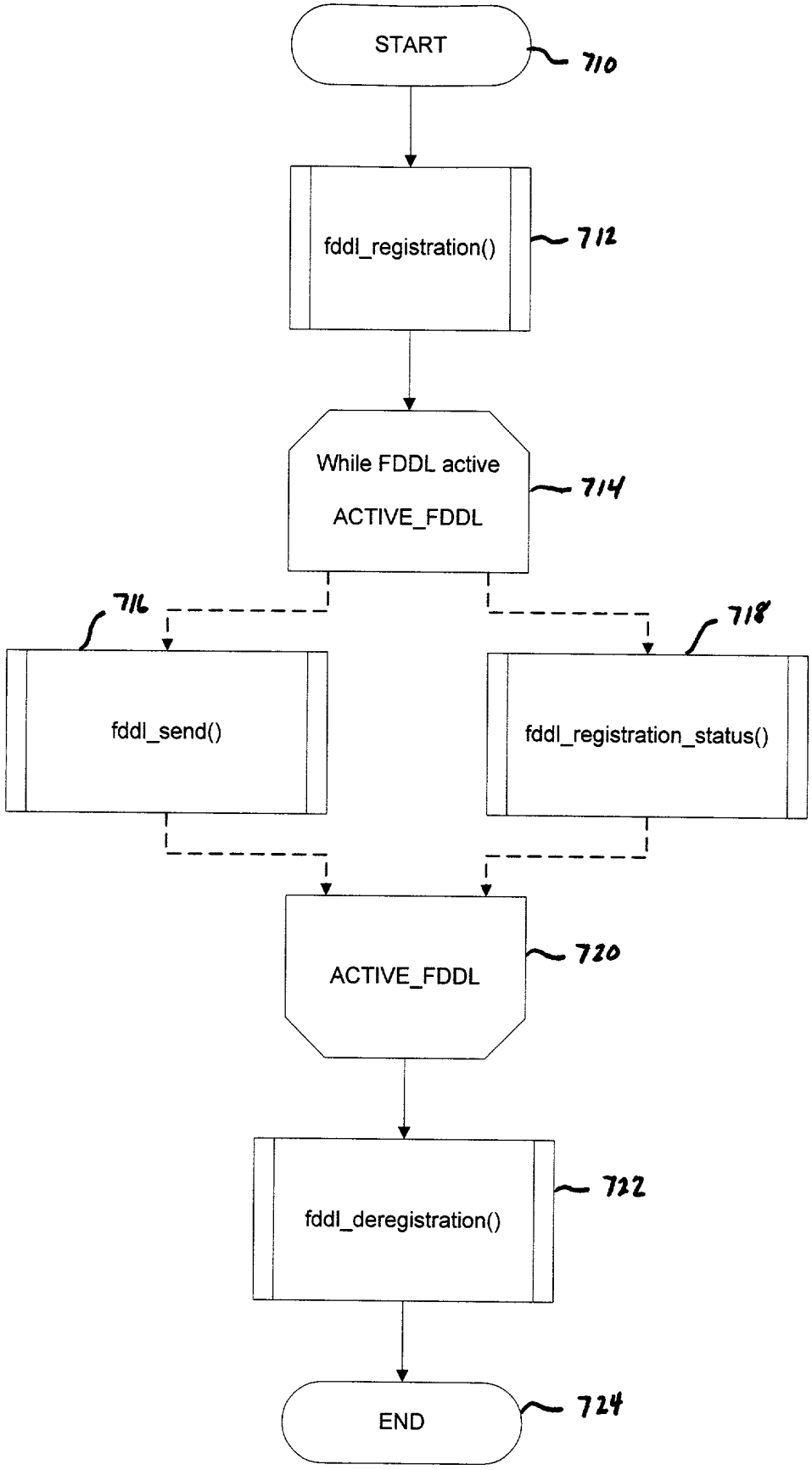
START
710
fddl_registration()
712
While FDDL active
ACTIVE_FDDL
714
716
718
fddl_send()
fddl_registration_status()
ACTIVE_FDDL
720
fddl_deregistration()
722
END
724

Fig. 7

*IBM Docket No. RAL9-1999-0073*

# *DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION*

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name; I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled:

**Portable Networking Interface Method and Apparatus for Distributed Switching System**

the specification of which is identified by the attorney (IBM) Docket Number appearing above.

I hereby state that I have reviewed and understand the contents of the above- identified specification, including the claims.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

| Number | Country | Day/Month/Year | Priority Claimed |
|---|---|---|---|

I hereby claim the benefit (a) under Title 35, United States Code, §119(e) of any U.S. application listed below and identified as a provisional application or (b) under Title 35, United States Code, §120 of any U.S. application listed below and not identified as a provisional application, and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior U.S. application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application

Prior U.S. Applications

| Serial No. | Filing Date | Status |
|---|---|---|

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: Daniel E. McConnell, Reg. No. 20,360; Kenneth A. Seaman, Reg. No. 28,113; Joscelyn G. Cockburn, Reg. No. 27,069; Gerald R. Woods, Reg. No. 24,144; John D. Flynn, Reg. No. 35,137; Horace St. Julian, Reg. No. 30,329; Joseph C. Redmond, Jr., Reg. No. 18,753; John E. Hoel, Reg. No. 26,279; Christopher A. Hughes, Reg. No. 26,914; and Edward A. Pennington, Reg. No. 32,588.

Send all correspondence to: Joscelyn G. Cockburn, IBM Corporation 972/B656; PO Box 12195; Research Triangle Park, NC 27709.

***First Inventor:*** **Cedell Adam Alexander, Jr.**

***Signature:*** Cedell Adam Alexander Jr. 2-24-00

***Date***

***Residence:*** 5509 Nob Hill Road
Durham, NC 27704

***Citizenship:*** USA

***Post Office Address:*** Same as above

***Second Inventor:*** **Loren Douglas Larsen**

***Signature:*** Loren Douglas Larsen 2/14/2000

***Date***

***Residence:*** 12479 S. W. Canvasback Way
Beaverton, OR 97007

***Citizenship:*** USA

***Post Office Address:*** Same as above